Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 157 (2006) 11–17
www.elsevier.com/locate/entcs

A Rule-based System for Web site Verification [1]

D. Ballis[a,2] J. García-Vivó[b,3]

[a] *Dip. Matematica e Informatica, Via delle Scienze 206, 33100 Udine, Italy.*

[b] *DSIC, Universidad Politécnica de Valencia, Camino de Vera s/n, Apdo. 22012, 46071 Valencia, Spain.*

Abstract

In this paper, we describe a system, written in Haskell, for the automated verification of Web sites which can be used to specify (partial) correctness and completeness properties of a given Web site, and then automatically check whether these properties are actually fulfilled. It provides a rule-based, formal specification language which allows us to define syntactic/semantic conditions for the Web site by means of a user-friendly graphical interface as well as a verification facility for recognizing forbidden/incorrect patterns and incomplete/missing Web pages.

Keywords: Web site Verification, Rewriting, Formal Methods.

1 Introduction

The management of a complex Web site is a nontrivial task, in which the problematics related to the verification and the correction of the (semistructured) data play a fundamental role. As a matter of fact, it is far simpler to discover inconsistent information on the Web than to find a well-maintained

[1] This work has been partially supported by the EU (FEDER) and the Spanish MEC under grant TIN 2004-7943-C04-02, by ICT for EU-India Cross Cultural Dissemination Project under grant ALA/95/23/2003/077-054, and by Generalitat Valenciana under grant GR03/025.

[2] Email:demis@dimi.uniud.it

[3] Email:jgarciavivo@dsic.upv.es

doi:10.1016/j.entcs.2005.12.041

Web site. We believe that formal methods can bring a relevant contribution, giving support to Automated Web site verification.

For instance, the system XLINKIT [3] allows one to check the consistency of distributed, heterogeneous documents as well as to fix the (possibly) inconsistent information. Its specification language is a restricted form of first order logic combined with Xpath expressions. [5] presents a framework for modeling Web interactions and a type system, which can be employed to catch errors in interactive Web programs. Other approaches employ model-checking [7,8,6] and trace equivalence tests [4] in order to formalize and verify safety properties related to the dynamic behavior of Web services (e.g. liveness and deadlock prevention properties, properties on Web service workflows, etc.).

In our previous work [2], we have described the system VERDI which provides a rule-based language for the specification and the verification of syntactic as well as semantic properties of collections of XML/XHTML documents. Specifically, the system is able to detect missing/incomplete Web pages w.r.t. a given formal specification. Therefore, in our case, formal methodologies are used to check static constraints, rather than dynamic properties of the Web sites.

This paper describes an evolution of the VERDI system which improves several aspects of the previous tool. Firstly, the new specification language does offer the expressiveness and the computational power of functions (which are modeled as term rewriting systems [9]) and is enriched by a new class of rules (i.e., *correctness rules*) in order to express properties for the detection of erroneous/forbidden information. Moreover, our verification methodology allows one to investigate both the syntax and the semantics of a Web site, while the typical validation against DTDs and XML Schemas can only check the syntactic layer of a given site. Secondly, a graphical interface has been developed which provides a more friendly use of the tool.

The system is based on the theoretical framework we proposed in [1]. We use rewriting-based technology both to specify the required properties and to formalize a verification technique, which is able to check them.

2 Web site denotation

In our framework, a *Web page* is either an XHTML or an XML document. Since Web pages are provided with a tree-like structure, they can be straightforwardly translated into ordinary terms of a given term algebra. Note that XML/XHTML tag attributes can be considered as common tagged elements, and hence translated in the same way. Therefore, *Web sites* can be represented as finite sets of (ground) terms. An example of such translation is given in

```
<members>                                members(
   <member status="professor">             member(status(professor),
        <name> mario </name>                       name(mario),
        <surname> rossi </surname>                 surname(rossi)
   </member>                                )
   <member status="student">               member(status(student),
        <name> giulia </name>                      name(giulia),
        <surname> verdi </surname>                 surname(verdi)
   </member>                                )
</members>                               )
```

Fig. 1. An XML document and its corresponding encoding as a ground term.

Figure 1.

3 Web specification language

A Web specification is a triple (R, I_N, I_M), where R, I_N, and I_M are finite set of rules. The set R contains the definition of some auxiliary functions which the user would like to provide, such as string processing, arithmetic, boolean operators, etc. It is formalized as a term rewriting system, which is handled by standard rewriting [9]. The rewriting mechanism allows one to execute function calls by simply reducing them to their irreducible form (that is, a term that cannot be rewritten any longer).

The set I_N describes constraints for detecting erroneous Web pages (*correctNess rules*). A correctness rule has the following form:

$$\mathtt{l} \rightharpoonup \mathtt{error} \mid \mathtt{C}$$

where $\mathtt{l}$ is a term, $\mathtt{error}$ is a reserved constant, and $\mathtt{C}$ is a (possibly empty) finite sequence of membership tests w.r.t. a given regular language [4] (e.g. $\mathtt{X} \in \mathtt{rexp}$), and/or equations over terms of the form $\mathtt{s} = \mathtt{t}$, where $\mathtt{s}$ and $\mathtt{t}$ are terms. For the sake of expressiveness, we also allow to write inequalities of the form $\mathtt{s} \neq \mathtt{t}$ in $\mathtt{C}$. Such inequalities are just syntactic sugar for $(\mathtt{s} = \mathtt{t}) = \mathtt{false}$. When $\mathtt{C}$ is empty, we simply write $\mathtt{l} \rightharpoonup \mathtt{error}$.

Informally, the meaning of a correctness rule

$$\mathtt{l} \rightharpoonup \mathtt{error} \mid \mathtt{X_1\ in\ rexp_1}, \ldots, \mathtt{X_n\ in\ rexp_n}, \mathtt{s_1} = \mathtt{t_1} \ldots \mathtt{s_m} = \mathtt{t_m}$$

is the following. Whenever an instance $\mathtt{l}\sigma$ of $\mathtt{l}$ is recognized in some Web page $\mathtt{p}$, where σ is a substitution, and

- each structured text $\mathtt{X_i}\sigma$, $\mathtt{i} = 1, \ldots, \mathtt{n}$, is contained in the language of the corresponding regular expression $\mathtt{rexp_i}$;
- (ii) each instantiated equation $(\mathtt{s_i} = \mathtt{t_i})\sigma$, $\mathtt{i} = 1, \ldots, \mathtt{m}$, holds;

[4] Regular languages are denoted by the usual Unix-like regular expression syntax.

then, Web page p is marked as an incorrect page. Moreover, the instantiated pattern $\mathtt{l}\sigma$ provides us the piece of the Web page p containing the wrong information. Thus, we are not only able to find the erroneous Web page, but also to precisely locate which part of the document we have to remove/modify.

The third set of rules I_M specifies some properties for discovering incomplete/missing Web pages (*coMpleteness rules*). A completeness rule is defined as $\mathtt{l} \rightharpoonup \mathtt{r}\ \langle \mathtt{q} \rangle$, where l and r are terms and $\mathtt{q} \in \{\mathtt{E}, \mathtt{A}\}$. Completeness rules of a Web specification formalize the requirement that some information must be included in all or some pages of the Web site. We use attributes $\langle \mathtt{A} \rangle$ and $\langle \mathtt{E} \rangle$ to distinguish "universal" from "existential" rules. Right-hand sides of completeness rules can contain function calls of functions which are defined in R. We assume that each function call can be evaluated to its irreducible form (i.e. normal formal) w.r.t. R. Besides, some symbols in the right-hand sides of the rules may be marked by means of the symbol ♯. Marking information of a given rule r is used to select the subset of the Web site in which we want to check the condition formalized by r. Intuitively, the interpretation of a universal rule $\mathtt{l} \rightharpoonup \mathtt{r}\ \langle \mathtt{A} \rangle$ (respectively, an existential rule $\mathtt{l} \rightharpoonup \mathtt{r}\ \langle \mathtt{E} \rangle$) w.r.t. a Web site W is as follows: if (an instance of) l is recognized in W, also (an instance of) the irreducible form of r must be recognized in *all* (respectively, *some*) of the Web pages which embed (an instance of) the marked part of r.

Example 3.1 Let R be a TRS defining function Nat(X), which converts a string X to a natural number, append(X, Y) which concatenates two strings, and add(X, Y) which sums two natural numbers. Let (R, I_N, I_M) be a Web specification where I_N and I_M are defined as follows:

member(name(X), surname(Y)) ⇀ ♯hpage(fullname(append(X, Y)),status) ⟨E⟩

hpage(status(professor)) ⇀ ♯hpage(♯status(♯professor), teaching)) ⟨A⟩

hpage(X) ⇀ error | X in [:TextTag:]* sex [:TextTag:]*

blink(X) ⇀ error

project(grant1(X), grant2(Y), total(Z)) ⇀ error | add(Nat(X), Nat(Y)) ≠ Nat(Z)

pub(year(X)) ⇀ error | X in [0-9]*, ≤ (Nat(X), 1999) = true

The given Web specification models some required properties for a Web site of a research group. The first two rules are completeness rules, while the remaining ones are correctness rules. First rule formalizes the following property: if there is a Web page containing a member list, then for each member, a home page should exist which contains (at least) the full name and the status of this member. The full name is computed by applying the function append to the name and the surname of the member. The marking information establishes that the property must be checked only on home pages (i.e., pages containing the tag "hpage"). Second rule states that, whenever a home page of a professor is recognized, that page must also include some teaching information. The rule is universal, since it must hold for each professor home page.

Such home pages are selected by exploiting the mark given on the tag "status". The third rule forbids sexual contents from being published in the home pages of the group members. Precisely, we check that the word `sex` does not occur in any home page by using the regular expression `[:TextTag:]* sex [:TextTag:]*`, which identifies the regular language of all the strings, built over the set of all the tags and raw texts, containing that word. The fourth rule is provided with the aim of improving accessibility for people with disabilities. It simply states that blinking text is forbidden in the whole Web site. The fifth rule states that, for each research project, the total project budget must be equal to the sum of the funds, which has been granted for the first and the second research periods. The sixth rule formalizes the condition that only recent publications are cited in the Web site (5 last years).

3.1 The verification methodology

Diagnoses are carried out by running Web specifications on Web sites. The operational mechanism is based on a novel rewriting-based technique called *partial rewriting*, which is able to extract a partial structure from a term, and then rewrite it. Roughly speaking, partial rewriting is a rewriting relation in which pattern matching is replaced by a *simulation* algorithm (cf. [1]).

In order to find correctness errors, we follow a method which is rather simple. We apply correctness rules to Web pages. If a Web page is partially rewritten to the constant `error`, then a correctness error for that page is raised, since a piece of erroneous/forbidden information has been recognized. As for completeness errors, we first generate a set of requirements (i.e. data which must be contained in the site) by partially rewriting the Web pages via the completeness rules, then we use a simulation-based algorithm to check whether the requirements are fulfilled, that is, the required information is not missing. When a requirement is not satisfied, it witnesses the lack of some information and the system outputs the incomplete Web page `p` together with the information which should be added to `p` in order to fulfill the requirement.

4 The GVerdi verification system

The verification system has been implemented in Haskell (GHC v6.2.2) and is publicly available together with a set of examples at

`http://www.dsic.upv.es/users/elp/GVerdi.`

The implementation consists of approximately 1100 lines of source code.

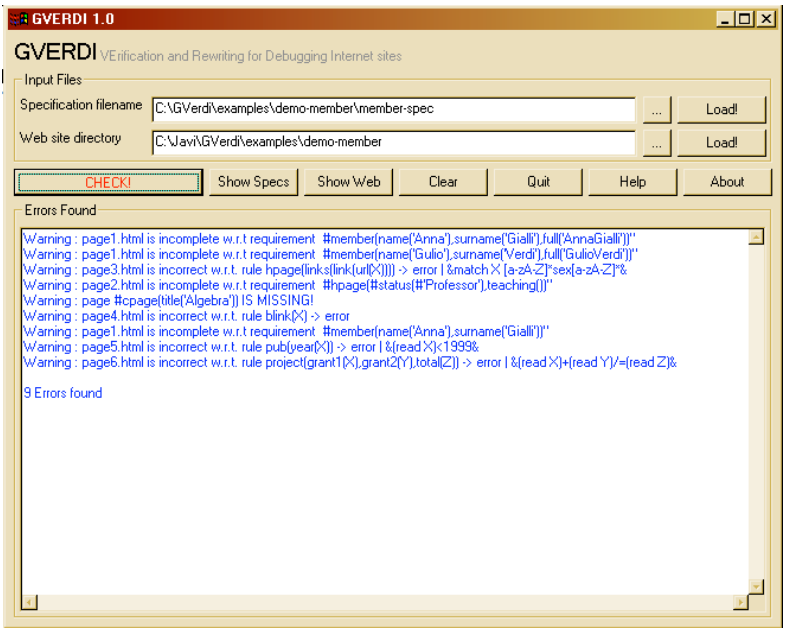

Fig. 2. Screenshot of the verification system GVerdi.

It includes a parser for semistructured expressions (i.e. XML/XHTML documents) and Web specifications, and several modules implementing the partial rewriting mechanism, the verification technique, and the graphical user interface. A snapshot of the running system is shown in Figure 2. The system allows the user to load a Web site together with a Web specification. Additionally, he/she can inspect the loaded data and finally check the Web pages w.r.t. the Web site specification. We have tested the system on several Web sites. In many cases, we were able to detect both missing/incomplete and incorrect Web pages efficiently.

References

[1] M. Alpuente, D. Ballis, and M. Falaschi. Automated Verification of Web Sites Using Partial Rewriting. In *1st Int'l Symposium on Leveraging Applications of Formal Methods (ISoLA'04)*, pages 81—88, 2004.

[2] M. Alpuente, D. Ballis, and M. Falaschi. VERDI: An Automated Tool for Web Sites Verification. In J. J. Alferes and J. Leite, editors, *Proc. of the 9th European Conference on Logics in Artificial Intelligence (JELIA'04)*, pages 726–729. Springer LNCS 3229, 2004.

[3] L. Capra, W. Emmerich, A. Finkelstein, and C. Nentwich. xlinkit: a Consistency Checking and Smart Link Generation Service. *ACM Transactions on Internet Technology*, 2(2):151–185, 2002.

[4] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based Verification of Web Service Compositions. In *Proc. of 18th IEEE International Conference on Automated Software Engineering (ASE'03)*, pages 152–163. IEEE Computer Society, 2003.

[5] P. T. Graunke, R. B. Findler, S. Krishnamurthi, and M. Felleisen. Modeling Web Interactions. In *Proc. of 12th European Symposium on Programming (ESOP'03)*, volume 2618 of *Lecture Notes in Computer Science*, pages 238–252. Springer, 2003.

[6] P. Inverardi and S. Scriboni. Connectors Synthesis for Deadlock-Free Component-Based Architectures. In *Proc. of 16th IEEE International Conference on Automated Software Engineering (ASE'01)*, pages 174–181. IEEE Computer Society, 2001.

[7] N. Kaveh and W. Emmerich. Deadlock Detection in Distribution Object Systems. In *ESEC / SIGSOFT FSE 2001*, pages 44–51, 2001. ISSN: 0163-5948.

[8] N. Kaveh and W. Emmerich. Validating Distributed object and Component Designs. In *Formal Methods for Software Architecture*, pages 63–91. Springer LNCS 2408, 2003.

[9] J.W. Klop. Term Rewriting Systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume I, pages 1–112. Oxford University Press, 1992.